The Maximum Score in Super Don Quix-ote

Keir, Paul

Published in:
xCoAx 2015: Proceedings of the Third Conference on Computation, Communication, Aesthetics and X

Published: 01/09/2015

Document Version
Publisher's PDF, also known as Version of record

xCoAx 2015
Computation
Communication
Aesthetics
and X

Glasgow
Scotland
2015.xCoAx.org

The Maximum Score in Super Don Quix-ote

Paul Keir
School of Engineering and Computing, University of the West of Scotland, UK
paul.keir@uws.ac.uk

Keywords: Videogames, Retrogaming, Laserdisc, Artificial Intelligence, Computer Vision, Easter Eggs

Arcade laserdisc videogames were pioneered by the original 1983 release of Dragon's Lair from Advanced Microcomputer Systems. Alas, the punishing gameplay mechanics of Dragon's Lair left many players frustrated. The 1984 laserdisc game, Super Don Quix-ote, from Japanese developer Universal, continued to employ a traditional animation technique, while including on-screen prompts, providing the player with a helpful indication of the correct response to each challenge. By completing Super Don Quix-ote, without loss of life, a maximum score of 636500 can be achieved through routine gameplay bonus mechanisms. Super Don Quix-ote, however, also includes undocumented support for alternative responses to the on-screen prompts. In the project described here, the open-source Daphne laserdisc emulator; along with Super Don Quix-ote software including a binary ROM image of the game itself and associated video files; are provided as input to a computer vision system, proving a perfect score of 776500 is possible; then confirmed by the human hand. A video of the full playthrough is available at https://youtu.be/ZpzWhfh92F4, and submitted to the Twin Galaxies gaming records organisation.

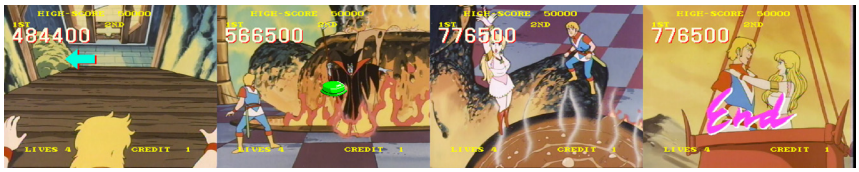

Fig. 1 Climactic scenes from Super Don Quix-ote (1984) courtesy of the Daphne Laserdisc Emulator

1 Introduction

During the golden era of arcade videogames, from the late 1970s to the late 1980s, the graphical content of games was technically, and arguably artistically, superior to that provided by home entertainment systems. Barring exceptions such as *Battlezone* (1980), *Cube Quest* (1983), or *I, Robot* (1983), arcade games were constructed using 2D sprite-based raster graphics; with noteworthy examples from 1984 including *Pac-Land*, *Marble Madness* and *Kung-Fu Master*.

The 1983 appearance of the arcade laserdisc title *Dragon's Lair* presented a step-change in the visual quality of arcade games. A Dragon's Lair player is presented with what appears to be a traditionally animated cartoon. The catch is that the delightful animation will soon end abruptly in the avatar's death, unless the player takes singular action at *precisely* the moment intended by the game designer. Such intermittent moments of challenge in videogames would ultimately become known as quick time events (QTE); after the relative success of Sega's *Shenmue* in 1999. An alarming aspect of the seminal QTEs in Dragon's Lair, however, is that they are *not* accompanied by an on-screen prompt. Consequently, the player must instinctively, and frequently, respond to the subtle and fleeting dangers embedded within the game; by one of four moves from the joystick, or a press of the button.

Super Don Quix-ote (SDQ) was released in 1984, and its Japanese heritage can be discerned in the anime character design and animation; and to a comparable degree, by its cheesy and bombastic American dub localisation. The significant gameplay difference in SDQ, is that the QTEs are accompanied by an on-screen prompt; one of four rather jarring blue arrow icons, inviting an up, down, left or right movement of the joystick; or a green button icon, prompting depression of the sole physical button.

Gameplay is supported by a damsel in distress story theme, wherein the eponymous hero must rescue his lady love from a demonic witch. References to the 17th century Spanish novel by Miguel de Cervantes, *The Ingenious Gentleman Don Quixote of La Mancha*, are minimal: a young Quixote retains the squireship of Sancho (Panza) and his donkey (Dapple); with the ingénue addressed as Isabella rather than Dulcinea. A notable windmill appears towards the end of the game, replete with giant.

1.1 Gameplay and Scoring Details

Typically a player who fails to respond to a QTE, or who responds incorrectly, will see the action cut to a scene involving Quixote's death; accompanied by a decrement in the lives tally. If lives remain, gameplay will resume at the start of a level which has yet to be completed. An ad-hoc score bonus is awarded immediately after each successful QTE; and a player completing a level with no loss of life, is awarded the sum of the score bonuses from that level. Having completed the entire game, a final bonus is rewarded, equal to; where is the number of lives remaining.[1] A player completing the game without loss of life can achieve a score of 636500. This is, however, not the maximum score possible.

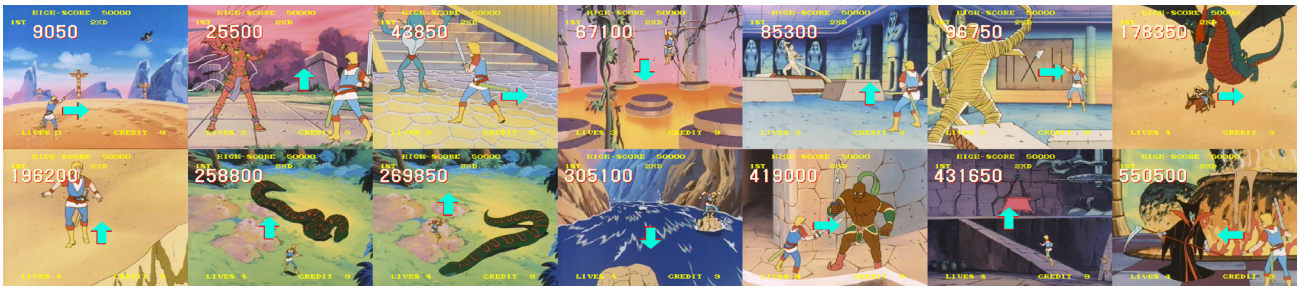

Fig. 2 The player is rewarded when using Button; Left; Button; Left; Left; Button; Down; Left; Left; Right; Left; Button; Left & Button (lexicographical ordering) instead of the on-screen prompts shown.

A selection of QTEs in SDQ allow responses distinct from those invited by the on-screen prompts. Figure 2 illustrates all fourteen such occasions. For example, the game's penultimate QTE prompt is shown bottom-right in Figure 2. An arrow invites a leftward movement of the joystick. Such a gesture is of course permitted; yet so is a *button press*. These 14 QTEs are sprinkled throughout the game, and each valid QTE alternate response provides the player an additional score bonus of 10,000. This project identifies *all* such QTE alternatives using custom software which exhaustively tries all possible responses to the 156 QTEs in SDQ. With this information, the game is subsequently completed by the author to obtain the maximum possible score in SDQ of 776500.

2 Software Development

To discover the full set of alternate QTE responses, an exhaustive automated search was planned. Two approaches then presented themselves: either modify the low-level source code of the Daphne (Ownby 2001) laserdisc emulator; or scan and analyse the display buffer using computer vision methods. With the source code for Daphne 32-bit, and the development system 64-bit Ubuntu, the first option was always on the back foot. The second approach was then selected; offering the attractive possibility to retarget software components towards other games or applications.

1 An extra life is awarded when a score of 100,000 is achieved. Play starts with 3 lives by default.

Obtaining a handle to the display buffer of the relevant X Window on Ubuntu is simplified by the libxdo library; readily available in the package manager as libxdo-dev. As libxdo is a C library, the inclusion of the xdo.h header must be guarded by the extern "C" linkage specifier. Listing 1 need then only check that a single matching window was found; `list[0]` provides that window handle.

Listing 1 Code to locate an X Window named "DAPHNE:"

```
xdo_t *xdo = xdo_new(NULL);
Window *list;
unsigned nwindows;
xdo_search_t search;
memset(&search, 0, sizeof(xdo_search_t));
search.max_depth    = -1;
search.require      = xdo_search_t::SEARCH_ANY;
search.searchmask |= SEARCH_NAME;
search.winname      = "DAPHNE:";
xdo_search_windows(xdo, &search, &list, &nwindows);
```

Having the Daphne emulator's X window, the libX11 library is used to obtain its width, height, and screen coordinates. A further library from the package manager, Imlib2, facilitates straightforward interaction with the display buffer. Assuming `x`, `y`, `w` and `h` hold the size and location information, Listing 2 demonstrates code to obtain a pointer, `data`, to a contiguous array of 32-bit Alpha-Red-Green-Blue (ARGB) data; `DATA32`. It is also straightforward to save this as an image file.

Listing 2 Code to obtain fast access to the X display buffer

```
Imlib_Image img = imlib_create_image_from_drawable(0,x,y,w,h,1);
imlib_context_set_image(img);
DATA32 const *data = imlib_image_get_data_for_reading_only();
```

2.1 Recognising On-Screen Prompts

The QTE on-screen arrow prompts are not subtle; but usefully, are comprised largely of a single shade of blue. Using the minimal grabc[2] tool, a mouse click will reveal that its hexadecimal ARGB representation is `0xff00ffd8j`[3]; while the arrow's red shadow is `0xfffd0100`. The four different QTE arrows can be recognised by traversal of the `data` array from Listing 2, looking for horizontal runs of blue pixels, followed by a shorter run of red; and vice versa. The green (`0xff00fe00`) on-screen button prompt is handled similarly.

2.2 Remote Control

With both the X window and a libxdo handle from Listing 1, commands to emulate the press and release of keys may be sent to the Daphne SDQ window as shown in Listing 3.

2 Grabc is also available from the Ubuntu package manager.

3 Different display drivers will generate different colour values.

Listing 3 Example of code which emulates the press and release of the right arrow key

```
xdo_send_keysequence_window_down(xdo, list[0], "Right", 0);
std::this_thread::sleep_for(std::chrono::milliseconds(100));
xdo_send_keysequence_window_up   (xdo, list[0], "Right", 0);
```

2.3 Knowing the Score

Alas, the score also requires comprehension. Analysis of the changing score can inform the algorithm as to whether an attempt at an alternate QTE response has been successful or not. An unchanging score can also evidence the loss of a life; and thankfully we can thereby avoid analysis of the life tally digits. The differences between score digits were more subtle than those between arrows, and a training set was obtained by hand; shown in Figure 3. Five horizontal scan lines were positioned to emphasise differences between the 10 digits. As with the on-screen prompts, the simplistic colour scheme of the score digits eased the matching process; though the white pixels of each digit do host minor variations, requiring a fuzziness in the matching of "white"; otherwise often `0xfffcffd9`.

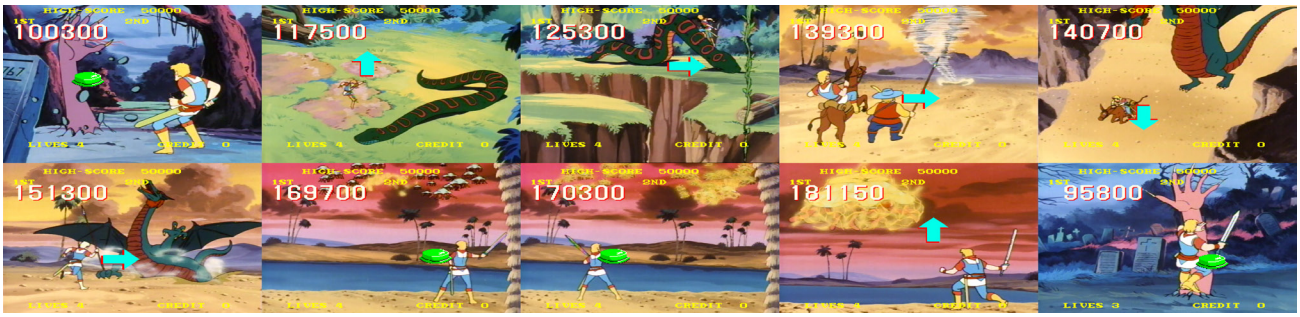

Fig. 3 The 10,000s in the screenshots above provide the digits 0-9 to assist score recognition

2.4 Unique Prompt Identification

The algorithm begins knowing the 156 conventional QTE responses. The DIP switches of SDQ, also supported by Daphne, allow an infinite lives[4] option. Infinite lives are useful in reducing the time required to search through all moves; an effect which becomes more pronounced as the game progresses. Nevertheless, the outside possibility of multiple correct alternate responses to a single QTE; and with that being the last QTE of a level, means that the algorithm must be capable of starting a new game; following the successful completion of the last one.

Unique identification of each on-screen prompt is complicated by the pseudo-random level order following a death event. Perceptual hashing was ruled out due to the observation that the lifetime of on-screen prompts may bridge a cut in the animation. Ultimately, it was discovered that the screen coordinates of the *first* on-screen prompt of a level, were unique. Knowing the

4 Infinite lives merely stops the lives tally (default is 3) falling, and has no further effect on scoring.

number of QTEs in a level, together with knowledge of the score, was sufficient to track and identify every QTE uniquely.

3 Conclusions and Future Work

A perfect score of 776500 for SDQ was obtained by locating all possible valid alternate QTE responses automatically using the computer vision methods outline above; informing a subsequent playthrough by the author; available on YouTube. Future work could ensure the program[5] can accommodate different pixel colours and display drivers; and also alternative screen resolutions, including full-screen. Such affairs are of course somewhat prosaic; the specific goal of the project has been achieved. A project which introduced the QTE icons of SDQ into Dragon's Lair through the Daphne emulator could potentially improve its gameplay, and build synergy between it and SDQ.

5 The C++ source code is available at https://bitbucket.org/pgk/sdq_explorer.

References

Matt Ownby. *DAPHNE Arcade Laserdisc Emulator. http://www.daphne-emu.com.* 2001.